On the Edge of Decision: Chaos Sensitivity as a Signal of LLM Reliability

Anonymous ACL submission

Abstract

This work explores how a large language model's (LLM) internal geometry and runtime sensitivity metrics can be used to determine whether a label will be incorrectly applied in a classification task. This work focuses on 3 distinct detection-based classification tasks: sentiment misclassification detection, prompt injection detection, and jailbreak detection. The purpose of this work is to train a classifier using a novel suite of metrics that can be collected during inference time. The resulting classifier outperforms baseline models and competes with models that have been specifically fine-tuned for the given task. By training a classifier on a suite of metrics inherent to a model's *internal geometries and layer-to-layer dynamics during inference, a tool for early misclassification detection that does not require ground truth labels, prompt engineering, or any preprocessing or postprocessing techniques can be developed.* The suite of metrics that are discussed in this work include Jacobian-derived scalars, representational geometry statistics, and decision boundary trajectories that occur during a single forward pass through the model's transformer layers. The prototype model outperformed baseline models, achieving CV-Accuracy scores of (0.952/0.911/0.874) and ROC scores of (0.977/0.969/0.949) across the 3 detection tasks.

1 Introduction

1.1 Problem Definition

LLMs are increasingly used in both enterprise and high-risk settings where prediction reliability is critical, yet *there is a lack of built-in systems to detect when an LLM is beginning to misalign with the task or when their outputs are likely incorrect.* This causes tremendous pitfalls in automation flows and can have compounding effects throughout a system's architecture that heavily relies on LLM output accuracy. Oftentimes a model may assign high probability to an incorrect answer due to pattern-matching artifacts in its training data rather than genuine reasoning about the input (Zhou et al., 2024; Tu et al., 2020; Du et al., 2023). It is also well known that LLMs have a tendency to hallucinate, causing an inherent misalignment between the given task and the expected output.

This work approaches the reliability problem at inference time by analyzing how hidden activations shape the geometric representation of a prompt during a single forward pass, how the decision boundary within that representation evolves across layers, and how sensitive predictions are to perturbations in layer-wise activations. *The central hypothesis is that correct and incorrect predictions leave distinguishable dynamical signatures in the network's layer-by-layer computation,* and the objective is to detect these signatures for use as an internal detection mechanism for the LLM.

1.2 Motivation

Inference-time monitoring and detection has several high impact applications. This section will discuss several real-world use cases.

1.2.1 Hallucination Detection

LLMs are widely known to hallucinate responses, yet they are often very convincing in the hallucinations they generate, oftentimes causing significant amounts of wasted time. This issue becomes especially problematic in domains requiring high factual accuracy, such as scientific research, where convincing hallucinations can reduce trust in model outputs and introduce substantial inefficiencies into downstream workflows.

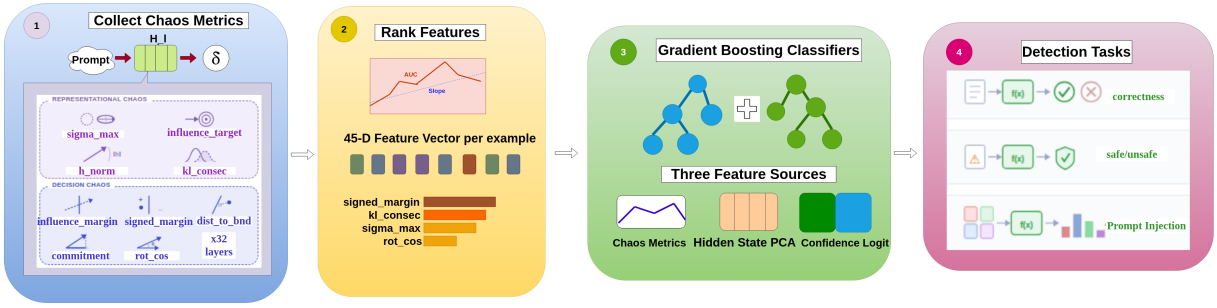


Figure 1: Chaos metric detection pipeline. (1) A single forward pass of a frozen LLM yields nine per-layer Jacobian metrics, split into representational and decision chaos families. (2) Each metric's depth trajectory is reduced to five aggregations (3 layer bands, slope, AUC) (3) Tree based classifiers are trained (4) The classifiers are applied to detection tasks (correctness, jailbreak, template),

When an LLM begins to form a hallucination in its hidden layers, the internal dynamics differ from those observed during a correct prediction (Duan et al., 2024; Ji et al., 2024). This provides an entry point to addressing the problem within the model itself rather than through ad hoc based approaches.

1.2.2 Prompt Injection

Bad actors have the capability to use adversarial prompts which cause the LLM to behave maliciously. This can occur by exploiting the model's innate tendency to follow instructions by embedding conflicting directives to the LLM, causing it to behave in ways that circumvents guardrails. This vulnerability is particularly devastating in systems that utilize retrieval-augmented and tool-augmented systems, where external content can be manipulated to influence the model's decision-making process (Greshake et al., 2023).

When models begin to deviate from their intended behavior during a hallucination, it is likely that the hidden state activation dynamics and geometries begin to follow a different trajectory (Chen et al., 2024). Because injected prompts may create detectable deviations in the model's internal sensitivity measurements, the classifier trained in this work has potential capabilities for detecting prompt injections at runtime.

1.2.3 Inference-Time Guardrailing

Current guardrail systems typically operate either as separate classifiers applied to the model's output text (Inan et al., 2023; Nandwana et al., 2025) or as a preprocessing step applied directly to the user's prompt (Muili, 2026; Alzahrani, 2026). *These strategies provide only surface-level detection and are prone to issues such as latency, prompt injection exposure, and a reduced robustness* when facing adversarial or rapidly evolving attack patterns (Zou et al., 2023).

The internal classifier referenced in this work, which has access to the model's internal computations, significantly reduces the feasibility of adversaries being able to circumvent this security mechanism.

1.3 Contributions

This work aims to advance inference-time model oversight by leveraging the evolving geometric and dynamical properties of transformer hidden states across the forward pass, *enabling the detection of model misalignment and other undesirable behaviors directly from internal representations rather than solely from generated outputs.*

The following 3 contributions are demonstrated in this work:

1. **Novel Jacobian based feature framework for inference time monitoring.** Nine metrics derived from Jacobian analysis and decision-boundary geometry are computed during a single forward pass and subsequently aggregated into a 45-dimensional feature vector. This novel feature vector is shown to carry meaningful signals to aid in misclassification, prompt injection, and jailbreak detection tasks, see Section 4.3.
2. **Trained gradient boosting classifier as a task-misalignment detection tool.** A

gradient boosting model is trained on a custom feature vector using contrastive token-based labels to define a binary classification objective. The classifier, trained on the novel feature framework introduced in this work, serves as a proof-of-concept for the utility of the proposed metric suite.

3. **Demonstrate the effectiveness of the detection classifier relative to baseline and task-specific models.** The proposed methodologies are evaluated against standard baseline approaches commonly used in prior work, as well as state-of-the-art models fine-tuned for prompt injection and jailbreak detection tasks.
   The results show that the proposed classifier consistently outperforms baseline models across all three tasks and achieves performance comparable to several domain-specific fine-tuned classifiers. A more detailed analysis is provided in Section 5.

2 Related Work

2.1 Hidden States and Hallucinations

Prior work has shown that the final hidden states of LLama-2 diverge measurably between correct and hallucinated answers, and PCA can be used to recover which directions in hidden space truthfulness exists (Duan et al., 2024). Another line of research investigates hallucination detection through the extraction of hidden state features that are predictive of hallucinations across three distinct stages of the forward pass: understanding, query processing, and generation (Wang et al., 2025).

The work in this paper is inspired by this prior research and aims to expand beyond the notion of reading from the resting positions of hidden states. Rather, this work analyzes the computational dynamics of a forward pass by measuring the evolving geometries and layer-to-layer trajectories of the hidden states that produced them. Notably, the static hidden state representation favored by prior work (Duan et al., 2024; Chen et al., 2024; Wang et al., 2025) appears in these experiments only as a baseline to be improved upon by the methodologies introduced in this paper, shown in Section 4.5.

2.2 Edge of Chaos

Prior work in dynamical systems and neural computation has suggested that optimal computation emerges near the edge of chaos, where systems balance stability and expressivity (Bertschinger et al., 2004). Recent studies extend this hypothesis to transformer-based language models, demonstrating improved reasoning performance near regimes that are between chaotic and stable states (Zhang et al., 2025b) and identifying chaotic characteristics in layer-wise reasoning dynamics using Lyapunov-inspired analyses (Li et al., 2025).

This work is heavily inspired by prior research examining transformer architectures through the lens of dynamical systems operating at the edge of chaos. Building on this perspective, the present work extends transformer-based dynamical system analysis to classification tasks and further investigates how decision boundaries and hidden state geometries evolve throughout the system.

2.3 Jacobian Analysis

Studying transformer architectures from a Jacobian derived perspective has given meaningful insights into how trajectories form within an LLM and how this gradient-based analysis impacts response generation and classification performance. In particular, studies examining transformer block Jacobians and their spectral properties have shown strong correlations between Jacobian structure, trajectory evolution, and downstream classification performance (Aubry et al., 2025; Saratchandran and Lucey, 2026; Lutz et al., 2026).

The experiments presented in this work attempt to *reduce familiar Jacobian geometries and layer-wise trajectory information into scalar values that form a novel feature vector*, which is hypothesized to carry a signal that can be used in different detection-based domains.

3 Methodology

This section describes the theory of the Jacobian and the 9 core metrics that constitute the proposed novel feature vector. These metrics are divided into two categories: *representational metrics* and *decision metrics*. The 4 representational metrics characterize the geometry and stability of hidden state evolution

across the transformer layer stack, while the 5 decision metrics quantify the relationship between the model's internal state and a classification boundary. This boundary is defined by the logit margin between two contrastive answer tokens (e.g. ["Yes", "No"], ["safe", "unsafe"]), which also serve as target labels during classifier training.

3.1 Jacobian

Most metrics in this work are derived from the Jacobian, which characterizes the local sensitivity of a transformer layer by describing how perturbations in the layer's input propagate to the layer's output hidden state representation (Hoffman et al., 2019). Predictions are restricted to the last token in the prompt (first token in the response), reducing the full Jacobian from $\mathbb{R}^{T\times d}$ to $\mathbb{R}^d$, an assumption used throughout this discussion.

For a transformer layer function f_l operating on hidden state h_l, the Jacobian is defined as:

$$J = \frac{\partial f(h)}{\partial h} \in \mathbb{R}^{d\times d} \tag{1}$$

Each entry J_{ij} captures the sensitivity of output dimension i with respect to input dimension j. Explicit computation of J requires $O(d)$ backward passes and results in an $O(d^2)$ memory cost, making the full Jacobian infeasible for inference-time use.

Therefore, directional sensitivities are computed using vector–Jacobian products (VJPs), which evaluate derivatives along chosen directions without forming J. For a vector $v \in \mathbb{R}^d$, this is given by:

$$J^\top v = \frac{\partial(v^\top f(h))}{\partial h}. \tag{2}$$

Thus, VJPs recover the required directional information using a single backward pass, avoiding full $O(d^2)$ materialization while retaining the sensitivity structure needed for the proposed chaos metrics.

3.2 Representational Metrics

The representational metrics characterize how hidden state trajectories evolve throughout the transformer during a forward pass. The 4 metrics used to capture representations and model's the internal dynamics of the model include damping behavior and stability measurements of hidden states, sensitivity to task-relevant directions, and the rate at which logit lens predictions converge across layers.

3.2.1 Jacobian Max-Singular Value

Each transformer layer is modeled as a differentiable function mapping a hidden state to a transformed representation. The local sensitivity of this function is captured by the Jacobian, and its spectral properties provide a means to quantify how perturbations are amplified by the layer.

Formally, the leading singular value of the Jacobian is defined by equation 4:

$$\sigma_1^{(l)} = \|J_l v^*\|_2, \tag{3}$$

where v^* is the dominant right singular vector of J_l. Rather than explicitly constructing J_l, v^* is approximated using power iteration, which repeatedly applies Jacobian–vector and vector–Jacobian products to converge to the dominant singular direction (Musco and Musco, 2015).

Algorithm 1 Power iteration with early stopping for estimating the leading singular value of the Jacobian.

Initialize random unit vector v_0
Set $\sigma_{\text{prev}} = 0$
for $i = 0$ to $K-1$ **do**
  Compute $u_i = J_l v_i$ (JVP)
  Compute $v_{i+1} = J_l^\top u_i$ (VJP)
  Normalize $v_{i+1} \leftarrow \frac{v_{i+1}}{\|v_{i+1}\|_2}$
  Compute $\sigma_i = \|J_l v_{i+1}\|_2$
end for
Return $\sigma_1^{(l)} = \sigma_i$

3.2.2 Influence Target

This metric is a sensitivity measure similar to the max singular value, but instead of capturing the most amplified Jacobian direction, it measures amplification specifically along the target logit direction. Specifically, it answers the question: *How much would an infinitesimally small perturbation to this layer's input effect the target logit's representation?* The target logit in this context is the ground truth label for any given example.

To extract this value at each layer, the head layer of the model is applied to the output of each hidden layer, yielding a logit vector of

shape $\mathbb{R}^{|\mathcal{V}|}$, where $\mathcal{V}$ is the model's vocabulary size.

Equation 4 below describes the influence margin metric.

$$I^{(l)}_{\text{target}} = \left\| \frac{\partial(\mathbf{v}_{\text{target}} \cdot \mathbf{y}_l)}{\partial \mathbf{h}_l} \right\|_2 \tag{4}$$

Where $\mathbf{v}_{\text{target}}$ is the row in the embedding vector that corresponds to the ground truth label, and y_l is the output of the hidden layer at the last token's position. The L2 norm of this gradient is the scalar value used in the framework.

3.2.3 Hidden state Norm

The hidden state of each layer in the model is a vector of shape $(, h_d)$. The L2 norm is simply the length of this vector and is given by equation 5 below. This metric is used to forage a signal that indicates the relative magnitude of the hidden state vector for each layer.

$$\|\mathbf{h}_l\|_2 = \sqrt{\sum_{i=1}^{d} h_{l,i}^2} \tag{5}$$

3.2.4 KL Divergence

The Kullback-Leibler (KL) Divergence refers to the softmax distribution of 2 consecutive hidden state vectors, which is given by equation 6 below.

$$D^{(l)}_{\text{KL}} = \sum_v P_l(v) \log \frac{P_l(v)}{P_{l+1}(v)} \tag{6}$$

The KL Divergence also requires the head layer of the model to be applied to the hidden state vector to obtain the logit vector. Softmax is then applied to get the probability distribution for the last token at the given layer. This captures a signal that answers how much the model's token prediction changes from layer to layer.

3.3 Decision Metrics

The 5 metrics categorized as decision-based metrics revolve around the existence of a decision boundary in the model's hidden state representations. This boundary is analogous to the hyperplane used in Support Vector Machines (SVMs) in which a hyperplane separates competing prediction directions. Unlike the fixed hyperplane in SVMs, the hyperplane in representational space rotates from layer to layer as the each layer contributes to the final prediction. Each of the metrics in this section quantify geometric and sensitivity-based relationships with this rotating boundary within hidden state space.

3.3.1 Influence Margin / Signed Margin

The influence margin is closely related to influence-target (Section 3.2.2) and measures how perturbations to a layer's input affect the margin between target and contrastive logits. It captures the sensitivity of the decision boundary to infinitesimal changes in the hidden representation, as shown in equation 7:

$$I^{(l)}_{\text{margin}} = \left\| \frac{\partial \left[(\mathbf{v}_{\text{target}} - \mathbf{v}_{\text{contrast}}) \cdot \mathbf{y}_l \right]}{\partial \mathbf{h}_l} \right\|_2 \tag{7}$$

The signed margin is defined as the projection of the hidden state onto the decision boundary normal, giving a signed distance in representation space where positive values indicate the target class and negative values indicate the contrastive class, given in equation 8:

$$m_l = (\mathbf{v}_{\text{target}} - \mathbf{v}_{\text{contrast}}) \cdot \text{RMSNorm}(\mathbf{h}_l) \tag{8}$$

RMSNorm ensures that representations remain comparable across layers and aligned with the normalization used during final decoding.

3.3.2 Distance to Boundary

In addition to raw distance from the boundary a current prediction lies, there exists a concept of fragility to the landscape in the current hidden state. To capture this concept of fragility, the distance to boundary metric divides the signed margin, or raw distance, by the gradient norm of the hidden state, as shown in equation 9 below.

$$d_l = \frac{|m_l|}{\|\nabla_{\mathbf{h}_l} m_l\|_2} \tag{9}$$

This metric captures that a large signed margin does not necessarily indicate a confident prediction, as it may result from instability in the current representational shift far from the decision boundary. This metric answers the following: *How much would the representation need to change before the model switches to the alternative prediction?*

3.3.3 Rotation Cosine

The margin boundary's gradient vector points in a direction that maximizes the distance between the target and contrastive logits at each layer. Because this direction evolves from layer to layer, it is necessary to capture the degree of rotation at each layer, formalized in equation 10. This vector represents the direction that most increases prediction confidence and is equivalent to the cosine similarity of the gradient vectors of layer l and $l+1$.

$$\cos\theta_l = \frac{\mathbf{g}_l \cdot \mathbf{g}_{l+1}}{\|\mathbf{g}_l\| \, \|\mathbf{g}_{l+1}\|} \tag{10}$$

3.3.4 Commitment

Each transformer layer adds a residual update $\Delta\mathbf{h}_l = \mathbf{h}_{l+1} - \mathbf{h}_l$ to the hidden state. Commitment measures whether this update actually moved the hidden state in the direction that reinforces the current prediction and is given by equation 11 below:

$$c_l = (\mathbf{h}_{l+1} - \mathbf{h}_l) \cdot \hat{\mathbf{g}}_l \tag{11}$$

where $\hat{\mathbf{g}}_l = \mathbf{g}_l / \|\mathbf{g}_l\|$ is the unit margin gradient. This metric answers the question: *did this layer actually move toward confidence?*, and captures the intent of each hidden layer's change to the residual stream.

4 Experimental Setup

4.1 Overview

All experiments were run using Meta Llama-3.1-8B-Instruct (Meta AI, 2024), an open-weight model with 32 transformer layers and hidden dimension size $d = 4,096$ and 32 attention heads. All data collection was performed on a single node equipped with two NVIDIA H100 NVL GPUs.

4.1.1 Datasets

Three binary classification tasks are evaluated: sentiment misclassification detection, prompt injection detection, and jailbreak detection. For sentiment analysis, 500 examples are sampled from the `IMDB` dataset (Maas et al., 2011) and evaluate predictions over the token set *["positive", "negative"]*. Each example is evaluated using four prompt templates, yielding 2,000 total evaluations; to account for template-induced correlation, GroupKFold is used with $k = 5$. The prompt injection dataset `deepset/prompt-injections` (deepset, 2023b) is merged across splits, producing 662 examples evaluated over the token set *["yes", "no"]*, with standard 5-fold cross-validation. For jailbreak detection, a balanced dataset of 158 examples is constructed from `rubend18/ChatGPT-Jailbreak-Prompts` (RubenD18, 2023) and benign prompts from `fka/awesome-chatgpt-prompts`, again using 5-fold cross-validation and the token set *["safe", "unsafe"]*.

4.2 Feature Aggregation

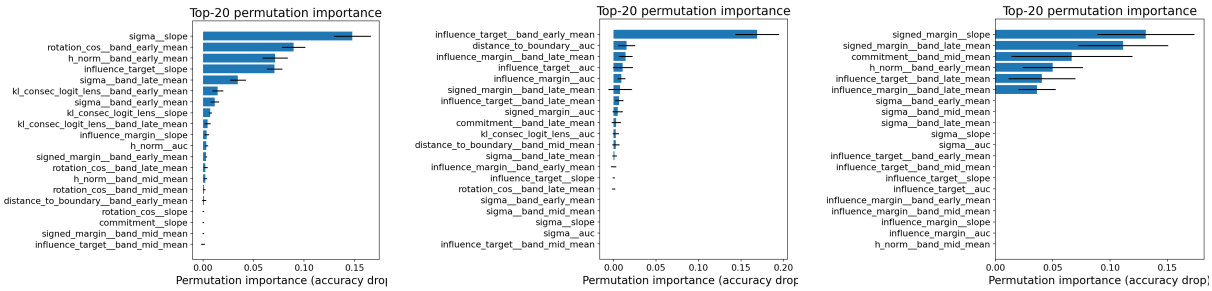
Each of the 9 metrics that constitute the novel feature vector were measured per layer. This meant $9 \times 32 = 288$ natural features are collected in this dataset. Training a classifier on a feature vector of this size would almost certainly lead to overfitting due to the curse of dimensionality, which spawns a need simplify the feature set, leading to the following aggregation process.

The mean of each metric was calculated across three bands: early layers (1–15), middle layers (16–28), and late layers (29–32). Late layers are only the last 3 layers due to the known phenomenon and observations of the final layers in LLMs contribute disproportionately to changes in predictions as the residual stream approaches the unembedding head (Meng et al., 2023).

Additionally, cumulative metrics of slope and area under the curve (AUC) is calculated for each metric across all 32 layers for a single forward pass. This results in 5 different bands for each of the 9 feature metrics, yielding $9 \times 5 = 45$ distinct features in the final feature vector.

4.3 Permutation Importance

The permutation importance concept is borrowed to demonstrate which features of the aggregated feature set carry a meaningful signal that can be used in classification. Each feature's importance is measured by how much the prediction accuracy shifts when a selected feature undergoes a series of random, small perturbations. *The higher the change in accuracy resulting from these permutations, the more important the feature (Breiman, 2001).* Figure 2 shows that the metrics carrying the strongest signal differ across tasks, with each task exhibiting its own clear set of standout metrics



(a) Sentiment (b) Prompt Injection (c) Jailbreak

Figure 2: Permutation importance (measured as the drop in accuracy after shuffling a metric, averaged over 30 repeats) identifies which metrics carry the strongest signal for each task. Across the three tasks, the dominant metric shifts by application: σ_1 slope is most important for sentiment misclassification detection, influence-target early-band mean for prompt injection detection, and signed margin slope for jailbreak detection. While the strongest metric differs across tasks, each task exhibits a clear set of high-signal metrics within the suite.

within the suite. For these results, n_repeats was set to 30, meaning 30 tiny permutations were applied to each metric.

This step is purely done as a statistical analysis step, to ensure there is in fact meaningful signals within the feature vector. This could also be used as a preprocessing step to prune the feature vector of less meaningful metrics, however this is left to future work, see Section 7.

4.4 Gradient Boosting Classifier

Because the training data for the classifier prototype is inherently tabular, that is, it contains mixed-scaled, well-structured data, a Gradient Boosting (GB) model was selected to perform classification on the prototype model and PCA baseline models. The Histogram GB (HGB) variant was specifically chosen for optimizations relating to how the feature set is binned.

4.5 Baseline Models

Two different baseline models were used for comparison with the prototype model: a hidden state PCA baseline and a pure logit output-confidence baseline.

4.5.1 Hidden State PCA

The chaos metrics that compose the feature set in the prototype model essentially summarize hidden state sensitivities and geometries, so the question arises, *"Why not use the raw hidden states themselves?"* This baseline does exactly this, but instead of using the $num_layers \times h_d$ number of features, the large hidden state vector is PCA'd down to the same dimensionality of the prototype's feature vector for fair comparability. These are similar concepts to the prior work discussed in Section 2.1.

4.5.2 Logit Output Confidence

This baseline takes the simple approach of using the model's own token probability distribution across the token sets as a correctness predictor. This naive baseline explores the inherent ability of the model to perform these detection tasks, and acts as a demonstration tool for why tuning the HGB classifiers are necessary.

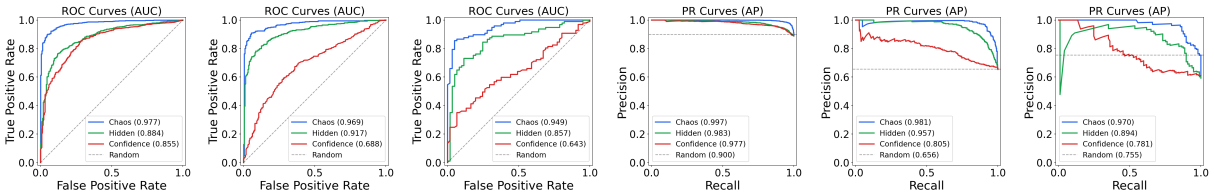
5 Results

All classifiers are evaluated using three complementary metrics:

Task	Model	Acc	ROC	PR
Sent.	Chaos	**.952**	**.977**	**.997**
	PCA	.897	.884	.983
	Confidence	.592	.855	.977
P. Inj.	Chaos M.	**.911**	**.969**	**.981**
	PCA	.846	.917	.957
	Confidence	.630	.688	.805
	SOTA (Ivry and Nahum, 2025)	.987	—	—
	DeBERTa-v3 (deepset, 2023a)	.991	—	—
Jailbreak	Chaos M.	**.874**	**.949**	**.970**
	PCA	.779	.857	.894
	Confidence	.588	.643	.781
	Snow.+RF (Galinkin and Sablotny, 2024)	.973	—	—
	JBShield-D (Zhang et al., 2025a)	.980	—	—
	LGuard-3-8B (Inan et al., 2023)	.953	—	—

Table 1: Classifier comparison across tasks together with specialized task-specific systems. Fine-tuned in-distribution models achieve the highest task-specific scores, while chaos metrics remain competitive without task-specific training.

- **CV Accuracy**: Mean accuracy across 5 K-Fold splits. Provides an interpretable measure of overall discrimination but can be dominated by the majority class.



(a) Sent. ROC (b) P. Inj. ROC (c) Jailbreak ROC (d) Sent. PR (e) P. Inj. PR (f) Jailbreak PR

Figure 3: ROC curves are sharply concave, quickly rising toward the top-left corner for Chaos (blue) and Hidden (green), while Confidence (red) follows a near-diagonal trajectory, reflecting weaker discrimination for true positives. PR curves maintain high precision at low recall before tapering as recall increases, indicating that the classifiers are most accurate on the clearest positive examples and become less reliable as they attempt to capture harder cases.

- **ROC-AUC**: Area under the Receiver Operating Characteristic curve. Measures discrimination across all threshold settings, robust to class imbalance, which is important as the models can experience up to an 89% correct prediction imbalance.
- **PR-AUC**: Area under the Precision-Recall curve. Particularly informative for the minority class (incorrect predictions), as it directly measures the ability to identify incorrect outputs with high precision.

5.1 Task Comparison

Table 1 presents the full results across all three tasks as well as a comparison to state-of-the-art models in their respective task domains. The chaos metrics classifier consistently outperforms both baselines on every task and every metric, and compares to some of the weight-adjusted fine-tuned models.

The CV Acc. and ROC scores demonstrates the largest gaps between prototype and baselines - Sentiment (Acc./ROC = 0.952/0.977), Prompt Injection (0.911/0.969), and Jailbreak (0.874/0.949). The higher ROC values indicates that the prototype exceeds baseline abilities to assign higher probabilities to positive examples and lower probabilities to negative examples.

Higher CV accuracy indicates more frequent correct predictions, with Chaos metrics outperforming PCA by 0.116 on average (0.055 Sentiment, 0.065 Prompt Injection, 0.095 Jailbreak) and Output Confidence by 0.300 on average (0.360, 0.281, 0.286).

Additionally, the precision-recall scores generally outperform the baseline models across all 3 tasks, with Chaos metrics outperforming PCA by 0.016 on average (0.014 Sentiment, 0.024 Prompt Injection, 0.076 Jailbreak) and Output Confidence by 0.036 on average (0.020, 0.176, 0.189). This indicates the prototype model is also generally better at identifying the positive class without making false positive mistakes.

See figure 3 for task-wise comparisons of the evaluation metrics.

6 Conclusions

The experiments in this work indicate that a prototype model trained on Jacobian-derived metrics, namely chaos metrics, outperforms baseline models. Additionally, the model is comparable to state-of-the-art fine-tuned models trained on task-specific data, which is typically far more computationally expensive. The prototype model's strong cross-task performance suggests that chaos-driven metrics are consistent in capturing relevant signals for detecting LLM misalignment.

Because the hidden-state PCA baseline draws on the same internal representations and still trails by 6-9 accuracy points on every task, the predictive signal cannot be attributed to where activations land in representation space, but rather it seems to reside in the dynamics and properties captured by the novel feature set.

The most critical component of this methodology realization that the suite of chaos metrics can be collected at inference time, and only a simple HGB model is necessary for healthy detection accuracies to exist. This approach differs from higher-performing models in that it does not require access to the inference model's weights. This methodology only requires access to metrics that can be probed and calculated at runtime, and results can be given before generation ever occurs.

7 Limitations

There are several limitations to this experiment that future work could expand on. Currently, the experiments are setup to predict only the last token in the prompt, or first token in the response generation. Additionally, the prompts are constructed to expect an output that is binary in nature, which is very different from a typical response that samples a token from the entire vocabulary. This limitation exposes a very clear boundary that needs to be crossed if this methodology were to be used for real world misalignment detection where the response tokens not only spans the entire vocabulary of a model, but also generates sequences of tokens rather than just a single token at a time.

Another pain point exists in the data that is used to train the prototype chaos model. That is, the chaos metrics that are collected in these experiments are shown to be overkill from permutation importance, and many of the can be pruned out before training the model as only the top metrics from permutation importance carry any meaningful signal. It is left to future work to develop optimizations in this area.

The chaos metrics mentioned in this work are by no means complete. Although not included in this report due to incomplete implementation, work has begun in exploring additional measurements of hidden-state geometry and sensitivity. These include Lyapunov exponents, which quantify the divergence rate of inter-layer predictions and statistics, and Fisher Information Matrix (FIM)-based metrics, which capture second-order curvature in high-dimensional parameter space (Soen and Sun, 2021; Sipio, 2025), beyond the first-order gradient information provided by the Jacobian. One of the biggest limitation of taking measurements in parametric space is the amount of computational required to form such high dimensional matrices, meaning certain approximation algorithms would be necessary.

Some of the datasets are small, especially the jailbreak detection dataset. More research could be done to find more comprehensive benchmark datasets in these domains. Additional work can be done to use a interchangeable prompt templates to build synthetic data if needed.

Only Llama-3.1-8b-Instruct was used as the inference time model. This methodology needs to test model robustness before any significant claims can be made about the metrics generally describing LLM hidden state behaviors. Llama-3.1-8b-Instruct is a comparatively small model to today's standards, so caution needs to be taken when scaling this methodology to larger models where inference times may increase dramatically. It is left to future work to reproduce this methodology into larger reasoning models.

8 Ethical Considerations

Because this work explores inference-time monitoring strategies, care should be taken in how the internal geometries are probed and how the internal states of the models are protected when considering any type of real world implementation of these strategies. An understanding of the ability of these internal probings to provide adversarial information for bad actors is required before any attempts of use for these methodologies. It is entirely possible that the metrics introduced in this work could provide insights into optimizing adversarial prompts or reverse-engineering internal model behavior for malicious intent, so proper safeguarding measures should always be taken when collecting these metrics.

A second consideration concerns the context in which possible deployments may exist. Because the tasks evaluated in these experiments are relevant to prompt injection and jailbreak detection, both relevant to security-based applications, it is critical to understand possible failure cases. The prototypes and referenced models in this paper do not give perfect accuracy scores, so it is important to consider how fatal false positives and/or false negatives have on the application or deployment used with the findings in this paper.

Acknowledgments

Gnerative AI tools were used to assist with language refinement and grammatical issue checking during manuscript preparation. Generative AI was also used to support code implementation under close human supervision and verification. Additionally, Gen AI tools were during the literature survey process by identifying potentially relevant publications. All references found with the support of Gen AI tools were

verified to be authentic, relevant sources. The authors reviewed and validated all AI-assisted outputs and take full responsibility for the contents of this paper. All scientific ideas, analyses, and conclusions presented are original to the authors.

References

Ahmed Alzahrani. 2026. Promptguard: A structured framework for injection-resilient language models. *Scientific Reports*, 16(1277).

Murdock Aubry, Haoming Meng, Anton Sugolov, and Vardan Papyan. 2025. Transformer block coupling and its correlation with generalization in llms. *Preprint*, arXiv:2407.07810.

Nils Bertschinger, Thomas Natschläger, and Robert Legenstein. 2004. At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. Inside: Llms' internal states retain the power of hallucination detection. *Preprint*, arXiv:2402.03744.

deepset. 2023a. deberta-v3-base-injection. Hugging Face Model Card. Accessed: 2026-05-24.

deepset. 2023b. Prompt injections dataset. `https://huggingface.co/datasets/deepset/prompt-injections`. Hugging Face Dataset Repository.

Mengnan Du, Fengxiang He, Na Zou, Dacheng Tao, and Xia Hu. 2023. Shortcut learning of large language models in natural language understanding. *Preprint*, arXiv:2208.11857.

Hanyu Duan, Yi Yang, and Kar Yan Tam. 2024. Do llms know about hallucination? an empirical investigation of llm's hidden states. *Preprint*, arXiv:2402.09733.

E. Galinkin and J. Sablotny. 2024. Improved llm jailbreak detection via pretrained embeddings. *arXiv preprint arXiv:2412.01547*.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. 2023. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. *Preprint*, arXiv:2302.12173.

Judy Hoffman, Daniel A. Roberts, and Sho Yaida. 2019. Robust learning with jacobian regularization. *Preprint*, arXiv:1908.02729.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *Preprint*, arXiv:2312.06674.

Dror Ivry and Oran Nahum. 2025. Sentinel: Sota model to protect against prompt injections. *Preprint*, arXiv:2506.05446.

Ziwei Ji, Delong Chen, Etsuko Ishii, Samuel Cahyawijaya, Yejin Bang, Bryan Wilie, and Pascale Fung. 2024. Llm internal states reveal hallucination risk faced with a query. *Preprint*, arXiv:2407.03282.

Xiaojian Li, Yongkang Leng, Ruiqing Ding, Hangjie Mo, and Shanlin Yang. 2025. Cognitive activation and chaotic dynamics in large language models: A quasi-lyapunov analysis of reasoning mechanisms. *Preprint*, arXiv:2503.13530.

Patrick Lutz, Themistoklis Haris, Arjun Chandra, Aditya Gangrade, and Venkatesh Saligrama. 2026. Layerwise dynamics for in-context classification in transformers.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023. Locating and editing factual associations in gpt. *Preprint*, arXiv:2202.05262.

Meta AI. 2024. Meta llama 3 8b instruct. `https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct`. Hugging Face Model Repository.

Olanrewaju Muili. 2026. Evaluating hybrid guardrail architectures for prompt injection defense in large language models. SSRN.

Cameron Musco and Christopher Musco. 2015. Randomized block krylov methods for stronger and faster approximate singular value decomposition. *Preprint*, arXiv:1504.05477.

Mahesh Kumar Nandwana, Youngwan Lim, Joseph Liu, Alex Yang, Varun Notibala, and Nishchaie Khanna. 2025. Taxonomy-adaptive moderation model with robust guardrails for large language models. *Preprint*, arXiv:2512.05339.

RubenD18. 2023. Chatgpt jailbreak prompts dataset. https://huggingface.co/datasets/rubend18/ChatGPT-Jailbreak-Prompts. Hugging Face Dataset Repository.

Hemanth Saratchandran and Simon Lucey. 2026. Spectral conditioning of attention improves transformer performance. *Preprint*, arXiv:2603.07162.

Riccardo Di Sipio. 2025. Rethinking llm training through information geometry and quantum metrics. *Preprint*, arXiv:2506.15830.

Alexander Soen and Ke Sun. 2021. On the variance of the fisher information for deep learning. *Preprint*, arXiv:2107.04205.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An empirical study on robustness to spurious correlations using pre-trained language models. *Preprint*, arXiv:2007.06778.

Peiran Wang, Yang Liu, Yunfei Lu, Jue Hong, and Ye Wu. 2025. What are models thinking about? understanding large language model hallucinations "psychology" through model inner state analysis. *Preprint*, arXiv:2502.13490.

Shenyi Zhang, Yuchen Zhai, Keyan Guo, Hongxin Hu, Shengnan Guo, Zheng Fang, Lingchen Zhao, Chao Shen, Cong Wang, and Qian Wang. 2025a. Jbshield: Defending large language models from jailbreak attacks through activated concept analysis and manipulation. *Preprint*, arXiv:2502.07557.

Shiyang Zhang, Aakash Patel, Syed A Rizvi, Nianchen Liu, Sizhuang He, Amin Karbasi, Emanuele Zappala, and David van Dijk. 2025b. Intelligence at the edge of chaos. *Preprint*, arXiv:2410.02536.

Yuhang Zhou, Paiheng Xu, Xiaoyu Liu, Bang An, Wei Ai, and Furong Huang. 2024. Explore spurious correlations at the concept level in language models for text classification. *Preprint*, arXiv:2311.08648.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *Preprint*, arXiv:2307.15043.